Altibase Tools & Utilities

Migration Center User's Manual

Release 6.6

September 30, 2013

ALTIBASE

Altibase Tools & Utilities Migration Center User's Manual

Release 6.6

Copyright © 2001~2011 Altibase Corporation. All rights reserved.

This manual contains proprietary information of Altibase® Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright, patent and other intellectual property law. Reverse engineering of the software is prohibited.

All trademarks, registered or otherwise, are the property of their respective owners.

Altibase Corporation

10F, Daerung PostTower II, 182-13,

Guro-dong Guro-gu Seoul, 152-847, South Korea

Telephone: +82-2-2082-1000 Fax: 82-2-2082-1099

Homepage: http://www.altibase.com

Contents

Preface ..... **i**
About This Manual ii
Types of Users ii
Software Dependencies ii
How This Manual is Structured ii
Documentation Conventions iii
Related Documents v
Online Manuals vi
Altibase Welcomes Your Comments vi
1. Introduction **1**
1.1 Overview 2
1.2 System Requirements 3
1.2.1 Hardware requirements 3
1.2.2 Software requirements 3
1.2.3 Compatible Database Systems 3
1.3 Installation and Uninstallation 5
2. Getting Started **7**
2.1 Understanding Migration Center 8
2.1.1 Terminology 8
2.1.2 Basic Concept 8
2.2 Graphic User Interface (GUI) Mode 10
2.2.1 Understanding User Interface 10
2.2.2 Quick Guide for GUI Mode 11
2.3 Command Line Interface (CLI) Mode 14
2.3.1 Understanding CLI mode 14
2.3.2 Executing the Run Step in CLI mode 14
2.4 Tools 15
2.4.1 PSM Converter for File 15
2.4.2 Zip to Report 15
3. Migration Center Internals **17**
3.1 Build Step 18
3.1.1 Purpose 18
3.1.2 Output 18
3.1.3 Internal Activities 18
3.2 Reconcile Step 19
3.2.1 Purpose 19
3.2.2 Output 19
3.2.3 Internal Activities 19
3.3 Run Step 22
3.3.1 Purpose 22
3.3.2 Output 22
3.3.3 Internal Activities 22
AppendixA. Migration Options **25**
DB to DB Migration Options 25
DB to File Migration Options 26
AppendixB. Migratable Database Objects **27**
Other Databases to ALTIBASE HDB 27
AppendixC. Data Type Mapping **29**
Manipulating Data Type Mapping 29
Default Data Type Mapping Tables 29
Oracle Database to ALTIBASE HDB 29
Microsoft SQL Server to ALTIBASE HDB 31
MySQL to ALTIBASE HDB 32
Informix 11.5 to ALTIBASE HDB 34
AppendixD. PSM Converter Rule List **37**
View Conversion Rules 37

RULE-11001 37
RULE-11002 38
RULE-11003 38
Trigger Conversion Rules 38
RULE-12002 38
RULE-12003 39
RULE-12004 39
RULE-12005 40
RULE-12006 40
RULE-12007 41
RULE-12008 41
Function Conversion Rules 42
RULE-13001 42
RULE-13002 42
RULE-13003 42
RULE-13004 43
RULE-13005 43
RULE-13006 44
RULE-13007 44
Procedure Conversion Rules 45
RULE-14001 45
RULE-14002 45
Materialized View Conversion Rules 46
RULE-15004 46
Package Conversion Rules 46
RULE-16001 46
Library Conversion Rules 47
RULE-17001 47
RULE-17002 47
DML Conversion Rules 48
RULE-20001 48
RULE-20002 48
RULE-20003 49
RULE-20004 49
RULE-20005 50
RULE-20006 50
RULE-20007 51
RULE-20008 51
RULE-20009 52
RULE-20010 52
RULE-20011 52
RULE-20012 53
RULE-20013 53
RULE-20014 54
RULE-20015 54
RULE-20016 55
RULE-20017 55
RULE-20018 56
RULE-20019 56
RULE-20020 57
RULE-20021 57
RULE-20023 57
RULE-20024 58
RULE-20025 58
RULE-20026 59
RULE-20027 59
RULE-20028 59
RULE-20029 60
PSM Conversion Rules 60

RULE-30001 60
RULE-30002 61
RULE-30003 61
RULE-30004 62
RULE-30005 63
RULE-30006 64
RULE-31001 64
RULE-31002 65
RULE-31003 65
RULE-31004 66
RULE-31005 66
RULE-31006 67
RULE-31007 68
RULE-31008 68
RULE-31009 69
RULE-32001 69
RULE-32002 70
RULE-32003 71
RULE-32004 72
RULE-32005 72

Preface

About This Manual

This manual describes how to use Migration Center to perform database migration to ALTIBASE HDB™.

Types of Users

This manual has been prepared for the following ALTIBASE HDB users:

- Database administrators
- Performance managers
- Database users
- Application developers
- Technical support workers

It is recommended that those reading this manual possess the following background knowledge:

- Basic knowledge of computers, operating systems, and operating system commands
- Experience in using relational databases and an understanding of database concepts
- Experience in database server, operating system or network administration

Software Dependencies

This manual has been prepared assuming that ALTIBASE HDB 6 will be used as the database server.

How This Manual is Structured

This document is an elementary guide for those unfamiliar with Migration Center. The rest of this document is organized as follows:

- Chapter1: Introduction

  This chapter is intended to introduce Migration Center to all users who want to understand its features and install it.

- Chapter2: Getting Started

  This chapter first introduces basic Migration Center concepts in order to help users run Migration Center more efficiently and effectively. It then describes in detail how to use Migration Center in GUI and CLI modes.

- Chapter3: Migration Center Internals

  This chapter describes the Build, Reconcile, and Run migration steps in detail.

- Appendix A. Migration Options

This appendix explains the Migration Center properties.

- Appendix B. Migratable Database Objects

  This appendix contains a table that shows what database objects can be migrated using Migration Center.

- Appendix C. Data Type Mapping

  This appendix explains how to check and customize the default data type mapping for an existing project. Finally, it provides default data type mapping tables.

- Appendix D. PSM Converter Rule List

  This appendix lists and explains rules of SQL Converter.

Documentation Conventions

This section describes the conventions used in this manual. Understanding these conventions will make it easier to find information in this manual and in the other manuals in the series.

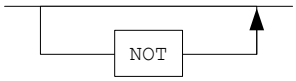
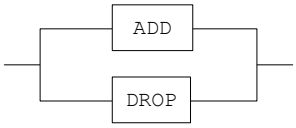
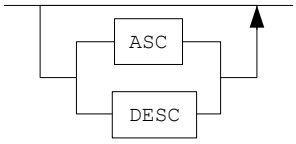
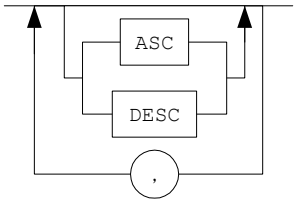
There are two sets of conventions:

- Syntax Diagrams
- Sample Code Conventions

Syntax Diagrams

This section defines and illustrates the format of commands that are available in Altibase products. These commands may include alternative elements, as well as required and optional parts.

Element	Description
	Indicates the start of a command. If a syntactic element starts with an arrow, it is not a complete command.
	Indicates that the command continues to the next line. If a syntactic element ends with this symbol, it is not a complete command.
	Indicates that the command continues from the previous line. If a syntactic element starts with this symbol, it is not a complete command.
	Indicates the end of a statement.

Element	Description
	Indicates a mandatory element.
	Indicates an optional element.
	Indicates a mandatory element comprised of options. One, and only one, option must be specified.
	Indicates an optional element comprised of options.
	Indicates an optional element in which multiple elements may be specified. A comma must precede all but the first element.

Sample Code Conventions

The code examples explain SQL statements, stored procedures, iSQL statements, and other command line syntax.

The following table describes the printing conventions used in the code examples.

Convention	Meaning	Example
[]	Indicates an optional item.	`VARCHAR [(size)] [[FIXED |] VARIABLE]`

Convention	Meaning	Example
{}	Indicates a mandatory field for which one or more items must be selected.	{ ENABLE | DISABLE | COMPILE }
|	A delimiter between optional or mandatory arguments.	{ ENABLE | DISABLE | COMPILE } [ENABLE | DISABLE | COMPILE]
. . .	Indicates that the previous argument is repeated, or that sample code has been omitted.	iSQL> select e_lastname from employees; E_LASTNAME ------------------------ Moon Davenport Kobain . . . 20 rows selected.
Other symbols	Symbols other than those shown above are part of the actual code.	EXEC :p1 := 1; acc NUMBER(11,2);
Italics	Statement elements in italics indicate variables and special values specified by the user.	SELECT * FROM *table_name*; CONNECT *userID*/*password*;
Lower Case Letters	Indicate program elements set by the user, such as table names, column names, file names, etc.	SELECT e_lastname FROM employees;
Upper Case Letters	Keywords and all elements provided by the system appear in upper case.	DESC SYSTEM_.SYS_INDICES_;

Related Documents

For additional technical information, please consult the following manuals:

- ALTIBASE HDB Installation Manual
- ALTIBASE HDB Administrator's Manual
- ALTIBASE HDB Replication User's Manual
- ALTIBASE HDB Precompiler User's Manual
- ALTIBASE HDB ODBC User's Manual
- ALTIBASE HDB Application Program Interface User's Manual
- ALTIBASE HDB iSQL User's Manual
- ALTIBASE HDB Utilities Manual
- ALTIBASE HDB Error Message Reference

Online Manuals

Manuals are available from Altibase Customer Support (http://support.altibase.com/).

Altibase Welcomes Your Comments

Please feel free to send us your comments and suggestions regarding this manual. Your comments and suggestions are important to us, and may be used to improve future versions of the manual.

Please be sure to include the following information :

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your full name, address, and phone number

For immediate assistance regarding technical issues, please contact the Altibase Customer Support (http://support.altibase.com).

We always appreciate your feedback and suggestions.

1 Introduction

This chapter is intended to introduce Migration Center to all users who want to understand its features and install it. It contains the following:

1.1 Overview

1.2 System Requirements

1.3 Installation and Uninstallation

1.1 Overview

Migration Center is a database migration tool that copies commonly compatible database objects and data from other databases to ALTIBASE HDB, either directly or indirectly. Most databases comply with international standards, but each database has its own tweaks which make manual database migration inevitable. Typically, manual database migration tasks are complicated, time-consuming, and prone to human error. Migration Center helps users process database migration with only a few mouse clicks in Graphic User Interface (GUI) mode. It also supports the execution of migration using a Command Line Interface (CLI) mode.

The major benefits of this tool are:

1. It can be used to migrate database objects from a source database to ALTIBASE HDB easily.

2. It can be used to copy data from a source database system to ALTIBASE HDB directly using a JDBC driver, which is the most popular and widely available interface for DBMS these days.

3. It can be used to export data to external files that can be imported into ALTIBASE HDB later using the iLoader tool.

4. It provides some parallel execution options to shorten the duration of the migration process. By default, it automatically balances the data migration load between a number of threads. It also gives users a way to manually split data between threads.

5. It provides data type mapping of different data types between heterogeneous databases. It supports default type mapping for convenience as well as user-defined data type mapping for maximum flexibility.

6. It provides not only a GUI mode for better usability, but also a CLI mode for special circumstances.

7. It allows users to modify DDL SQL texts for migrating database objects as Schema Migration.

As of version 6.6, Migration Center supports Oracle Database 10g or higher, Microsoft SQL Server 2005 or higher, MySQL 5.0 or higher and ALTIBASE HDB 4 or higher as the source database. Not all database objects can be handled using Migration Center, but these limitations will be eliminated in the near future.

1.2 System Requirements

This section sets forth the system prerequisites required to install and run Migration Center, and lists the database management systems with which Migration Center is compatible.

1.2.1 Hardware requirements

1.2.2 Software requirements

1.2.3 Compatible Database Systems

1.2.1 Hardware requirements

1.2.1.1 GUI Mode

- CPU: 800MHz Pentium III or better
- Main memory: 512MB or more
- Disk: 150MB or more free space
- Screen resolution: 1024 x 768 pixels or higher

1.2.1.2 CLI Mode

- CPU: 800MHz Pentium III or better
- Main memory: 512MB or more
- Disk: 150MB or more free space

1.2.2 Software requirements

- Java Development Kit (JDK) or Java Runtime Environment (JRE) 5 or higher

Migration Center is a pure Java application that uses Swing for the GUI mode. It mostly runs independently from the user's hardware and operating system, but relies on the JRE. We recommend installing Oracle or IBM Java 5 or higher. To run Migration Center in GUI mode, the user's environment must support Java Swing.

Migration Center is bundled with JRE 5 for 32-bit Microsoft Windows systems. Therefore, users using 32-bit versions of Windows don't need to take additional steps to provide the JRE. However, for other operating systems, users will be required to provide and configure the JRE and related environment variables appropriately.

1.2.3 Compatible Database Systems

1. ALTIBASE HDB

   a. As a source database: 4.3.9 or higher
   b. As a destination database: 5.5.1 or higher

2. Oracle Database: 10g or higher

3. Microsoft SQL Server: 2005 or higher

4. Oracle MySQL: 5.0 or higher

Migration Center can be used with some different versions of ALTIBASE HDB and other databases. It uses the JDBC driver to connect to database, so it is important to prepare the appropriate JDBC driver for the source and destination databases. For users' convenience, Migration Center comes with a few JDBC drivers for supported databases.

Please note that for legal reasons, the Microsoft SQL Server JDBC driver file and MySQL Connector/J file is not included in the package.

Please visit following sites to download appropriate JDBC driver file:

- SQL Server JDBC Driver 2.0 (a Type 4 JDBC Driver) for SQL Server 2008, 2005, or 2000

  http://www.microsoft.com/en-us/download/details.aspx?id=2505

- Microsoft JDBC Driver 4.0 (a Type 4 JDBC Driver) for SQL Server 2012, 2008 R2, 2008, 2005 or Microsoft SQL Azure

  http://www.microsoft.com/en-us/download/details.aspx?id=11774

- Oracle MySQL Connector/J

  http://dev.mysql.com/downloads/connector/j/

1.3 Installation and Uninstallation

Migration Center can be downloaded from Altibase Customer Center at http://support.altibase.com. Migration Center is distributed in zip and tar.gz file formats. The file contains executables and some JDBC drivers.

To install it, simply unzip the file. The resultant directory contains the "projects", "lib", "conf", and "images" subdirectories, other directories, and files, including migcenter.bat and migcenter.sh. Move this folder to a desired location. Migration Center is now ready to use.

To uninstall Migration Center, simply delete the Migration Center directory.

2 Getting Started

This chapter provides an introduction to the basic concepts of Migration Center to enable its efficient and effective use, a detailed description on its usage in GUI and CLI modes, and an overview of user convenient tools.

It is divided into the following four sections:

2.1 Understanding Migration Center

2.2 Graphic User Interface (GUI) Mode

2.3 Command Line Interface (CLI) Mode

2.4 Tools

2.1 Understanding Migration Center

This section is intended to introduce terminology and describe the overall process of working with Migration Center.

2.1.1 Terminology

2.1.2 Basic Concept

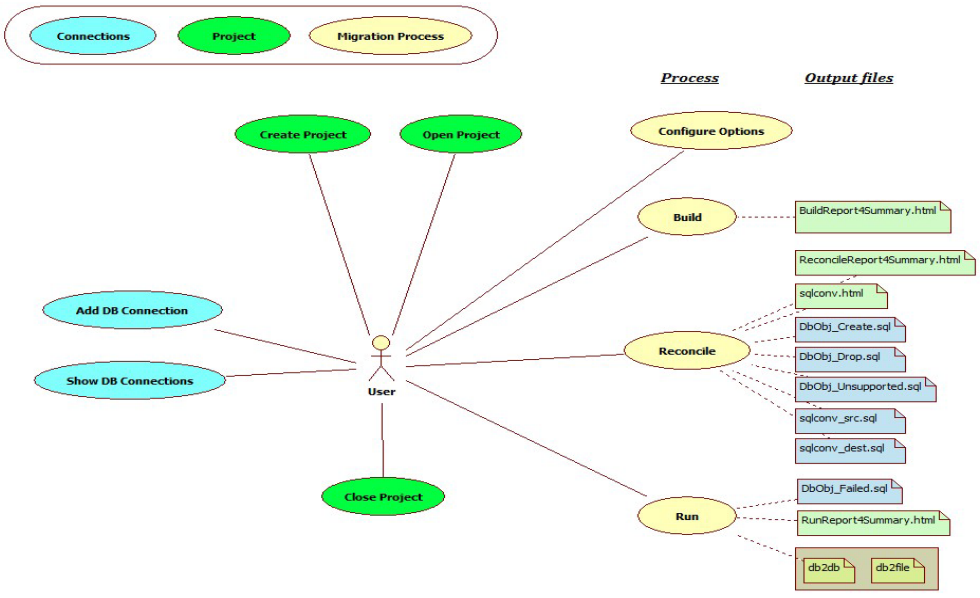
2.1.1 Terminology

2.1.1.1 Project

A Migration Center project is a fundamental work unit that describes all aspects of migration. It includes what to migrate, where to migrate it (i.e. from and to what kind of database or data file) and how to migrate database objects and the data in tables. Detailed information about Migration Options is provided in Appendix A. Migration Options. Only one project can be open at any time.

2.1.2 Basic Concept

The overall process of performing migration using Migration Center involves four steps: Prepare, Build, Reconcile and Run.



2.1.2.1 The Prepare Step

The Prepare step is an implicit step in the actual migration project. The final status of the Prepare step is having an open project with all database connections established. To get to the final Prepare step status, users may need to add working database connection information, open an existing

migration project, create a new migration project with preconfigured database connections, or establish connections with the databases.

2.1.2.2 Build Step

The Build step performs initial surveys on the current status of the source and destination databases. It retrieves information about the database objects in the source and destination databases through the open connections, and saves the information in the project directory. Since the information gathered in this step is used in other steps, it must reflect the current status. If any meta information is changed in the source database before the Run step, all of the steps from the Build step to the Run step must be re-executed in order to incorporate the changes.

2.1.2.3 Reconcile Step

The Reconcile step organizes a complete migration plan according to the current status and is mostly about making adjustments for differences, such as data types and tablespaces, between source and destination database systems. Parallel data migration and DDL SQL text editing can also be configured in this step. For instance, the user can specify which table of the source database is to be copied to which tablespace of the destination database, using how many work threads.

The user should keep in mind that this step is to be re-executed, if changes are made to the 'Migration Option'.

2.1.2.4 Run Step

The Run step executes the plan that was established in the Reconcile step. The schema and data are migrated, either directly or indirectly, in this step.

If the Migration Type in the Migration Options was set to “DB to DB”, then Migration Center creates the database objects in the destination database (schema migration). Then, it copies data from the source database to the destination database (data migration).

If the Migration Type in the Migration Options was set to “DB to File,” then SQL script files are created for migration, but the basic migration process is the same.

2.2 Graphic User Interface (GUI) Mode

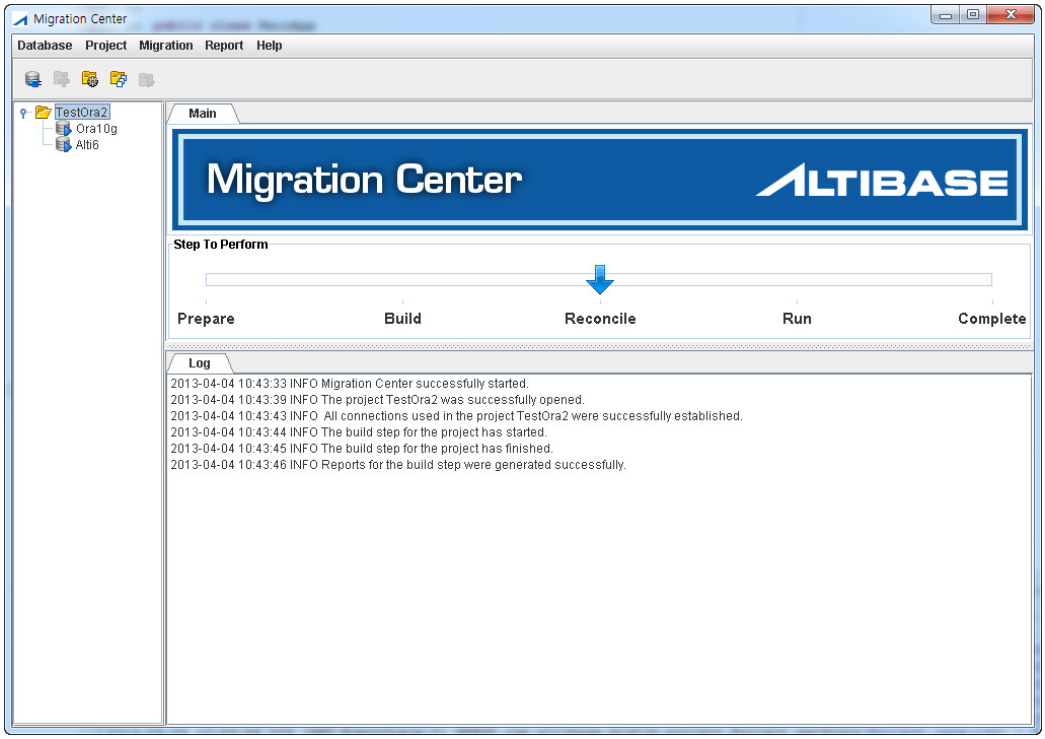
This section explains in detail how to use Migration Center in GUI mode. It starts by explaining the Migration Center user interface. It then provides quick step-by-step instructions on how to perform migration, and finally provides some more in-depth explanations of how to control Migration Center in GUI mode.

2.2.1 Understanding User Interface

2.2.2 Quick Guide for GUI Mode

2.2.1 Understanding User Interface

The Migration Center GUI consists of the three panes shown in the following picture. Starting in the top left-hand corner and moving clockwise, the three panes are the Project, Information, and Log panes.


Migration Center
Database Project Migration Report Help
TestOra2
Ora10g
Alti6
Main
Migration Center
ALTIBASE
Step To Perform
Prepare
Build
Reconcile
Run
Complete
Log
2013-04-04 10:43:33 INFO Migration Center successfully started.
2013-04-04 10:43:39 INFO The project TestOra2 was successfully opened.
2013-04-04 10:43:43 INFO All connections used in the project TestOra2 were successfully established.
2013-04-04 10:43:44 INFO The build step for the project has started.
2013-04-04 10:43:45 INFO The build step for the project has finished.
2013-04-04 10:43:46 INFO Reports for the build step were generated successfully.

2.2.1.1 The Project Pane

This provides a project-centric view. The Project pane shows the open project and its source and destination database connections. Only one project can be open in the project pane at a time.

2.2.1.2 The Information Pane

This pane provides a simple view for the project. It shows the current progress, and the remaining steps.

2.2.1.3 The Log Pane

This area outputs the result of actions performed by Migration Center in chronological order. It filters information that is of interest to the user from the logs stored in the "log" subdirectory under the directory in which Migration Center was installed.

2.2.2 Quick Guide for GUI Mode

This section is a quick step-by-step guide for new users. They will be able to perform migration easily in GUI mode simply by following the instructions specified in this section. The overall procedure is as follows:

1. Start Migration Center
2. Prepare a Project
    a. Register Source and Destination Database Connections
    b. Create a Project
    c. Open the Project
    d. Connect to Source and Destination Databases
    e. Set Migration Options
3. Build the Project
4. Reconcile the Project
5. Run the Project

2.2.2.1 Starting Migration Center

To run Migration Center in GUI mode in a Microsoft Windows environment, double click on the "migcenter.bat" icon in the Migration Center folder. In other operating systems, such as Linux, execute "migcenter.sh". If working in any environment other than a 32-bit Windows environment, be sure that the Java Runtime Environment (JRE) is installed, and that it is accessible from the terminal with the appropriate PATH environment setting.

2.2.2.2 Registering a Database Connection

1. Select the “Add Database Connection” menu item under the “Database” menu, or click on the “Add Database Connection” icon in the main toolbar.
2. Fill in all the input fields in the “Add Database Connection” dialog:
    a. DB Product: The name of the database system.
    b. Connection Name: A unique database connection name to be used for any Migration Center projects.
    c. IP: The Internet Protocol (IP) address of the database server.
    d. Port: The port number of the database server.
    e. User: A valid user ID for the database.
    f. Password: The password corresponding to the user ID for the database.
    g. JDBC Driver: The path to the JDBC driver file to be used to connect to the database.

h. Encoding: The default character set used on the client.
i. IP Version: The version of the IP protocol used in this connection.

3. Click the “Test” button at the bottom of the dialog to test the given information. This is required before Migration Center lets you add this connection to the connection list.

4. If the connection test is passed, click the "OK" button at the bottom of the dialog to add the connection to the connection list and close the “Add Database Connection” dialog.

2.2.2.3 Creating a Project

1. Select “Create Project” menu item under the “Project” menu, or click on the “Create Project” icon in the main toolbar.

2. Fill in the input fields in the “New Project” dialog:

   a. Project Name: A migration project name that is unique within Migration Center and meaningful to users.
   b. Project Path: The path to the folder in which the migration project files will be stored. This value is not be editable by users.
   c. Source Database: A database that contains the database objects to be migrated in this project.
   d. Destination Database: An ALTIBASE HDB database that will store the database objects from the source database. This input field is mandatory, even if the project is configured for “DB to File” migration.

3. Click the “OK” button at the bottom of the “New Project” dialog to dismiss it.

4. Users can continue to configure “Migration Options” by clicking the “OK” button in the information dialog.

2.2.2.4 Opening a Project

To open a project, select the “Open Project” menu item under the “Project” menu. This will show a simple file “Open” dialog. Users can simply choose a project folder and click the “Open” button.

2.2.2.5 Configuring the Project

Users can configure the basic strategy for the project before proceeding to set the project details. Please refer to Appendix A. Migration Options for a full explanation.

2.2.2.6 Connecting to Source and Destination Databases

Establishing source and destination database connections is essential in order for users to be able to proceed to the Build step for the project. To connect the databases for the project, select the “Connect” menu item under “Project” menu, or right-click on the open project and choose “Connect.”

2.2.2.7 Building the Project

The Build step is essential in the migration process, but simple to execute.

1. Select the “Build” menu item under the “Project” menu, or click on the “Build” icon on the main toolbar.

2. A “Build” dialog box will appear and show the build progress. When it is done, a “Report” button will appear at the bottom.

3. Click the “Report” button, and it will show a new dialog called “Build Report” and give you the option to copy the link to the HTML report file, or just close the dialog box. You can also accomplish this by selecting the “Build Report” menu item under the “Report” menu.

2.2.2.8 Reconciling the Project

The Reconcile step is very important to the migration process because it decides how the Run step executes, and it requires some knowledge of both source and destination databases. Detailed information about this procedure will be provided at 3 Migration Center Internals. Here are simple instructions:

1. Select the “Reconcile” menu item under the “Project” menu, or click the “Reconcile” icon on the main toolbar.

2. Choose either “User Mode” or “Table Mode” in the pop-up box.

3. If you chose “Table Mode” in the previous step, then choose the tables to be migrated from the source database.

4. Modify or approve the default data type mapping for your purpose.

5. Specify the default destination ALTIBASE HDB tablespaces for the tablespaces from the source database.

6. Modify or specify how any partitioned tables in the source database are to be migrated to ALTIBASE HDB.

7. Specify destination tablespaces for the individual tables in the source database.

8. Specify how the data migration load is to be distributed between migration threads.

9. Review DDL SQL texts for Schema Migration process.

10. Click the “Report” button to generate reports for this step.

2.2.2.9 Running the Project

Executing the Run step is as simple as building the project, but the effect is irreversible. Please be sure to check all settings before going any further.

1. Select the “Run” menu item under the “Project" menu, or click on the “Run” icon on the main toolbar.

2. Confirm that all configuration is correct by clicking “OK” on the warning dialog box.

3. Inspect the progress and final reports closely.

2.3 Command Line Interface (CLI) Mode

This section begins with an explanation of why it might be desirable to run Migration Center in CLI mode, and then explains how to do so. .

2.3.1 Understanding CLI mode

2.3.2 Executing the Run Step in CLI mode

2.3.1 Understanding CLI mode

GUI mode provides a user-friendly environment, but it is not always possible to run Migration Center in GUI mode. CLI mode comes in handy when GUI mode cannot be used, but it also has a limitation. CLI mode allows users to execute only the Run step of the migration procedure, because it's very inefficient to perform the Reconcile step at a terminal. Up to this point, CLI mode may have sounded like a complimentary mode to the GUI mode, and not of much interest, but it opens new possibilities when it comes to performance.

The main reason for using Migration Center in CLI mode is higher performance. Performing the Run step in CLI mode and GUI mode gives the same performance in same environment, but Migration Center can run in CLI mode in much tougher environments. The most time-consuming migration step among the four steps in commercial database environments is always the Run step. The essential job in the Run step is fetching data from the source database and passing it to the destination database. CLI mode can speed up data transfer if it resides on a server on which a GUI environment is not always available. It drastically reduces communication costs and reduces the total time required for migration.

The suggested workflow process is to complete the Prepare, Build, and Reconcile steps for a migration project in GUI mode, copy the entire Migration Center installation folder to the destination server, and then run the project in CLI mode.

2.3.2 Executing the Run Step in CLI mode

To execute the Run step of Migration Center in CLI mode, the command requires two arguments: *command* and *project_path*. The first argument is always "run". The second argument is the path, either relative or absolute, of the project to be built or executed.

```
sh > ./migcenter.sh command project_path
```

If either argument is omitted, Migration Center will attempt to start up in GUI mode.

2.4 Tools

This section introduces two utilities which are provided within Migration Center, and explains what they are and how to use them.

2.4.1 PSM Converter for File

"PSM Converter for File" enables the conversion of Oracle's PL/SQL to ALTIBASE PSM in independent file units. It reads a user-provided SQL file that contains at least one Oracle PL/SQL statement, translates it into ALTIBASE PSM, and then writes the result in a predefined SQL file, which is accompanied with a HTML-formatted report file. Internally, its functionality is equivalent to "PSM Converter" at the Reconcile step during Migration; for further information on converting rules, please refer to Appendix D. PSM Converter Rule List.

The user can run "PSM Converter for File" by clicking Tools -> PSM Converter for File on the main menu.

2.4.2 Zip to Report

"Zip to Report" is a utility that enables the user to easily report problems that arise at Migration Center. If the user runs this utility after opening a project, it collects problem tracking information, such as log files, revision number information, etc., and passes on this information as a zip file. The user only needs to send an email with the zip file attached to the Altibase Solution Center with his/her client registration information.

The user can run "Zip to Report" by clicking Tools -> Zip to Report on the main menu.

3 Migration Center Internals

This chapter discusses details of the three main steps of Migration Center, namely the Build, Reconcile, and Run steps. This chapter includes the following sections:

3.1 Build Step

3.2 Reconcile Step

3.3 Run Step

3.1 Build Step

3.1.1 Purpose

The Build step is not just about collecting information about database objects from the source and destination databases, but also about providing build reports so that users can perform migration more gracefully. The reports list migratable database objects in the source and destination database along with byte size information. This information will give users some idea about the scale of a migration project, and help to estimate the data volume and time required for the migration.

Please note that the information gathered in this step will be used throughout the entire migration process, and therefore it must reflect the current status of the source and destination databases. If there have been any changes to either the source database after Build step, then all steps must be re-executed.

3.1.2 Output

- Build reports: A number of data volume analysis reports on the current status of the source and destination databases, in HTML format and stored in the project folder.

3.1.3 Internal Activities

This step comprises two internal activities: collecting information about database objects from both databases, and generating Build reports based on the information.

Please consult 2.2.2.7 Building the Project to learn how to perform the Build step.

3.2 Reconcile Step

3.2.1 Purpose

The purpose of the Reconcile step is to create an organized plan for migration. Migration Center users should have their own plan for each database object. Migration Center cannot migrate all database objects into ALTIBASE HDB, but it can make migration easier by allowing full control over migration.

ALTIBASE HDB provides the benefits of an in-memory database for high performance and a disk based database for high capacity. The general strategy when using ALTIBASE HDB is to store frequently used and low-latency data in memory tablespaces, and to store the rest of the data in disk tablespaces. Please refer to the *Administrator's Manual* for more information about tablespaces in ALTIBASE HDB.

3.2.2 Output

- Reconcile reports: A number of reports that specify what to migrate and how to migrate database objects in the project folder.
- SQL Data Definition Language (DDL) scripts: A sample SQL file used to create and drop database objects in the destination database. It is provided for the user's benefit, but is not used in any of the Migration Center steps.

    1. DbObj_Create.sql: A SQL script file for creating database objects to be migrated.
    2. DbObj_Drop.sql: A SQL script file for dropping database objects having the same names as those to be migrated.
    3. DbObj_Unsupported.sql: A SQL script for creating unsupported database objects.

- PL/SQL Convert reports: A number of reports that are generated by PL/SQL Converter

    1. HTML format report:

        a. sqlconv.html: To compare any difference between source and converted PL/SQL
    2. Text format reports:

        a. sqlconv_src.sql: Input PL/SQL statements
        b. sqlconv_dest.sql: Converted PL/SQL statements with comments for applied rules

3.2.3 Internal Activities

Even though the Reconcile step is very important and can be complicated, its wizard-like UI makes it easy to follow. Please refer to the 2.2.2.8 Reconciling the Project section to learn how to start the Reconcile step.

The "User/Table Mode" dialog is the first dialog of this step, and determines the scope of migration. User mode includes all supported database objects for a particular user?? in the source database as migration targets, while Table mode allows particular tables and their dependent objects to be chosen as migration targets. Please refer to Appendix B. Migratable Database Objects for more details. If

you choose Table mode, then the next wizard dialog shows the “Set Target Tables” menu item at the top. If you choose User mode, then the next wizard dialog won’t show the “Set Target Tables” menu item at the top, but the rest of the menu items will be available, the same as when choosing Table mode.

A reconcile wizard dialog will appear after the “User/Table Mode” dialog. The wizard shows default settings and let you modify the settings. It guides you to each menu item in sequence, but you can choose to go a specific menu item by selecting it in the left pane.

The “Set Target Tables” menu item is only available if Table mode was selected in the previous step. Migration target tables can be selected using the “Add/Remove” buttons and subsequent dialogs.

The “Data Type Mapping” step is used to map data types between heterogeneous databases. Small differences between data types can cause unexpected data loss and truncation during data migration, so users should be aware of this possibility. Please refer to Appendix C. Data Type Mapping for more details.

The “PSM Data Type mapping” step is for mapping PSM data types between different databases. This step will be activated only for Oracle to ALTIBASE HDB PSM Migration. All configurations specified in this step will be reflected to “SQL Editing” step of PSM Migration later.

The “Tablespace to Tablespace Mapping” step is for mapping tablespaces between the source and destination databases. When a tablespace map is set, the contents of the tablespace are also mapped to the selected tablespace. This step creates a default tablespace map that can be altered using the “Table to Tablespace Mapping” menu item.

The “Table to Tablespace Mapping” step comprises mapping individual tables and indexes to tablespaces in the destination database by dragging and dropping them. Whenever the mapping is changed, the required total storage sizes for the relevant tablespaces in the destination database are recalculated. Internally, the sizes of database objects are kept in byte units for accuracy, but the unit of size shown in the dialog is MB, and the value is rounded. Therefore, the total size of a tablespace may not be the same as the total size of its contents.

The “Table to Thread Mapping” step introduce an automatic data load-balancing feature for improving the throughput of migration. The algorithm used for this feature splits the data in tables and then allocates them equally among data migration threads. However, this feature can be only applied to tables that have primary key or unique constraints. For cases in which a table cannot be migrated using this feature, this step also provides an alternative method for splitting the data in tables. This alternative method is manual but convenient.

The “SQL Editing” step provides users to review and modify DDL SQL texts for Schema Migration. User can directly modify DDL texts that will be applied to Destination DB by referencing Source DDL. SQL texts of creating Procedures, Functions, Triggers, and Views are displayed together in PSM Type. Users can choose Object Type to review by controlling Check Boxes in PSM Category. Those selected objects by controlling Check Boxes are displayed in either Done pane or To-Do List pane. Those objects displayed in To-Do List pane require users’ review. Those objects displayed in Done pane are compatible to ALTIBASE HDB and don’t require users’ review. The Source and Destination DDL of each object can be displayed by clicking each object names in each list. Users need to review Object DDL listed in To-Do List and click Save button. This will move the object from To-Do List pane to Done pane. It is possible even in objects in Done list can fail unexpectedly during Run Step. In this case, Users must review Missing Cause of Run Report, find out the cause of errors, and manually migrate those objects. Exporting any selected PSM Object DDL text as a text file is provided for users who prefer using their favorite text editor. This functionality is provided through Off-line Pane in PSM Object Type along with how to use that function. PL/SQL Converter embedded in Migration Center converts PSM type object DDLs to ALTIBASE HDB compatible object DDL. However, the con-

version is performed mostly on syntax. Users should be aware that semantic logic and contexts are not converted by this converter and they still require users' review.

Note: *Migration Center builds the internal dependency graph during Build step for Oracle to Altibase Migration. If users change these dependencies during editing Destination DDL, relevant objects cannot be guaranteed to be migrated.*

3.3 Run Step

3.3.1 Purpose

The purpose of the Run step is to copy database objects from the source database to the destination database (or to external files, depending on the Migration Options).

The results of this step are critical. The Run reports, which are generated after the Run step is complete, contain the result. The RunReport4Summary.html report file provides the overall result by comparing the number of database objects and table records in the source and destination databases. The RunReport4Missing.html report file describes the details of any failures.

The data that could not be migrated are collected in the "db2db" or "db2file" folder in the project folder. Those two folders have data files and form files which can be used with iLoader, the ALTIBASE HDB command-line data import/export tool. The "db2file" folder contains additional scripts and data files for importing data using iLoader. "iLoaderIn.sh" is the main script, and executes the other script files.

3.3.2 Output

- RunReport4Summary.html: The summary report file that provides the overall result of migration
- RunReport4Missing.html: The missing report file that provides information about data that could not be migrated, and the cause of each failure
- DbObj_Failed.sql: This contains a list of failed SQL commands and the cause of each failure.
- The db2db folder: A subfolder of the project directory in which data that could not be migrated are saved. It is only available when the "DB to DB" option is selected for "Migration Type" and "No" is selected for "Batch Execution."
- The db2file folder: A subfolder of the project directory in which all output is saved when the "DB to File" option is selected for "Migration Type"

3.3.3 Internal Activities

The Run step can be executed using a single mouse click in GUI mode or a single command in CLI mode. Please consult 2.2.2.9 Running the Project or 2.3.2 Executing the Run Step in CLI mode to learn how to perform the Run step.

Internally, this process comprises three steps: the PreSchema step, the Table and Data step, and the PostSchema step, to avoid complications related to database object dependencies. For instance, index objects are migrated during the PostSchema step, after the Table & Data step is completed, because it is faster to inserti data when no index is present than when an index is present. The detailed activities in each step are as follows:

1. PreSchema: Migrating sequence objects
2. Table & Data: Migrating table objects with content

3. PostSchema:

    a. Queue: Migrating queue objects
    b. Constraints: Migrating constraints, icluding as unique, primary key, foreign key and check constraints
    c. Index: Migrating index objects
    d. Synonym: Migrating private synonym objects
    e. Procedures, Functions, Materialized Views, Views, Typesets and Triggers: Depends on DBMS and version

Appendix A. Migration Options

Migration Options affect the migration project. They can be edited by selecting the "Migration Option" menu item from the Migration menu in GUI mode. Migration Options can usually be edited right after the project is created.

The primary option is the Migration Type: either "DB to DB" or "DB to File".

DB to DB Migration Options

DB to File Migration Options

DB to DB Migration Options

Name	Description
Migration Targets	The targets to be migrated: • Objects & Data: Database objects and table data • Objects: Database objects only • Data: Table data only
Execution Threads	The number of threads to be used to simultaneously migrate data.
Object Options: Foreign Key Migration	Specifies whether or not to migrate foreign key constraints.
Object Options: PSM Migration	Specifies whether or not to migrate procedures, functions, materialized views, views, typesets, and triggers.
Object Options: Drop Existing Objects	Specifies whether or not to drop and create target database objects on the destination database before copying data. If this option is set to "Yes", data of the tables of the destination database will be the same as that of the source database when migration is completed. If this option is set to "No", data from the tables of the source database will be inserted into the tables of the destination database after migration is completed.
Object Options: Keep Partition Table	Specifies whether or not to maintain partitioned tables.
Object Options: Use Double-quoted Identifier	Specifies whether or not to use double quotation marks for schema and object names.

Name	Description
Object Options: Postfix for reserved word	Specifies the user-defined word which is to be automatically postfixed to a database object name in the source database if it conflicts with a reserved keyword of ALTIBASE HDB.
Data Options: Batch Execution	Specifies whether or not to use JDBC batch insertion for higher performance.
Data Options: Batch Size	Specifies the batch size to be used for JDBC batch insertion.
Data Options: Commit Size	Specifies the commit size for JDBC insertion. Recommended values are equal to or greater than Batch Size or '0'. '0' means that commit is executed only after all of the data for each data insert thread has been inserted.
Data Options: Log Insert-failed Data	Specifies whether or not to log data that fails insertion during DB2DB data migration. This option is available only when the batch execution option is disabled.

DB to File Migration Options

Name	Description
Execution Thread	The maximum number of threads to use simultaneously to migrate data.
Migration Target	The targets to be migrated: • Objects & Data: Database objects and table data • Objects: Database objects only • Data: Table data
Object Options: Foreign Key Migration	Specifies whether or not to migrate foreign key constraints.
Object Options: PSM Migration	Specifies whether or not to migrate procedures, functions, materialized views, views, typesets, and triggers.
Object Options: Keep Partition Table	Specifies whether or not to change partitioned tables into non partitioned tables.
Object Options: Use Double-quoted Identifier	Specifies whether or not to use double quotation mark for schema and object names.
Data File: File Encoding	Specifies the encoding character set to be used for output scripts and data files.

Appendix B. Migratable Database Objects

Other Databases to ALTIBASE HDB

Database Object Type	Migratable At User Mode	Migratable At Table Mode	Note
Table	Yes	Yes	
Primary Key Constraint	Yes	Yes	
Unique Constraint	Yes	Yes	
Check Constraint	Yes	Yes	
Foreign Key Constraint	Yes	Yes	
Index	Yes	Yes	
Sequence	Yes	Yes	
Queue	Partly Yes	No	Supports only for ALTIBASE HDB as a source database
Private Synonym	No	No	
Procedure	Partly Yes	No	Supports only for Oracle and ALTIBASE HDB as a source database
Function	Partly Yes	No	Supports only for Oracle and ALTIBASE HDB as a source database
View	Partly Yes	No	Supports only for Oracle and ALTIBASE HDB as a source database
Materialized View	Partly Yes	No	Supports only for Oracle and ALTIBASE HDB as a source database
Trigger	Partly Yes	No	Supports only for Oracle and ALTIBASE HDB as a source database

Appendix C. Data Type Mapping

Migration Center's policy for mapping data types between heterogeneous databases is: "Try to minimize the loss of data". However, the situation may arise in which you wish to customize the way that data are mapped, even if it means the loss or corruption of data. To satisfy this requirement, Migration Center also provides a way to edit the data type mapping table.

This section explains how to check and customize the default data type mapping for an existing project. Finally, it provides default data type mapping tables.

Manipulating Data Type Mapping

Default Data Type Mapping Tables

Manipulating Data Type Mapping

You can check the default data type mapping and modify the way that data are mapped using the following methods:

1. Perform the Reconcile step for the project.
2. At the Data Type Mapping step, select the data type to manipulate and then click on the "Change" button.
3. Assign the new data type as desired and set precision and/or scale values.

Default Data Type Mapping Tables

Oracle Database to ALTIBASE HDB

Index	Source	Destination	Issue
1	CHAR	CHAR	

Index	Source	Destination	Issue
2	NCHAR	NCHAR	The explicit sizes of the source and destination NCHAR columns are the same, e.g. NCHAR(10) -> NCHAR(10). However, in the Oracle JDBC driver, the size of a national character column is defined as the number of bytes used, whereas in the JDBC driver of ALTIBASE HDB, the size of a national character column is defined as the number of characters that are stored. This means that the resultant column in ALTIBASE HDB will be two or three times as large as necessary, so please keep this in mind.
3	VARCHAR2	VARCHAR	
4	NVARCHAR2	NVARCHAR	The column sizes differ, for the same reason as NCHAR.
5	LONG	CLOB	
6	NUMBER	NUMERIC	
7	FLOAT	FLOAT	
8	BINARY FLOAT	FLOAT	
9	BINARY DOUBLE	VARCHAR (310)	There is no compatible data type in ALTIBASE HDB for the Oracle binary double type, so the data are stored in character form to prevent loss.
10	DATE	DATE	
11	TIMESTAMP	DATE	A small amount of data loss may occur due to the difference in scale. In Oracle, the scale of a timestamp value is nanoseconds (9 digits), whereas in ALTIBASE HDB, the scale of a timestamp value is microseconds (6 digits).
12	RAW	BLOB	
13	LONG RAW	BLOB	
14	BLOB	BLOB	
15	CLOB	CLOB	
16	NCLOB	NVARCHAR(10666)	There is no compatible data type in ALTIBASE HDB for the Oracle NCLOB, so the data are stored in NVARCHAR with the maximum precision. This may cause data loss during data migration when the actual data precision exceeds the NVARCHAR maximum size.

Microsoft SQL Server to ALTIBASE HDB

Index	Source	Destination	Issue
1	BIGINT	BIGINT	
2	DECIMAL	NUMERIC	
3	INT	INTTEGER	
4	NUMERIC	NUMERIC	
5	SMALLINT	SMALLINT	
6	MONEY	FLOAT	
7	TINYINT	SMALLINT	
8	SMALLMO-NEY	FLOAT	
9	BIT	CHAR(1)	
10	FLOAT	VAR-CHAR(310)	There is no compatible data type in ALTIBASE HDB for Microsoft SQL FLOAT type, so VARCHAR(310) is mapped to prevent data loss.
11	REAL	FLOAT	
12	DATE	DATE	
13	DATETIME2	DATE	A fraction of time can be loss due to difference in scale. Scale of DATETIME2 type of Microsoft SQL Server is hundreds of nanoseconds (7 digits), whereas the scale of DATE type of ALTIBASE HDB is only microseconds(6 digits).
14	DATETIME	DATE	
15	SMALLDATE-TIME	DATE	
16	CHAR	CHAR	
17	TEXT	CLOB	
18	VARCHAR	VARCHAR	
19	VAR-CHAR(MAX)	CLOB	
20	NVARCHAR	NVARCHAR	
21	NVAR-CHAR(MAX)	NVAR-CHAR(1066 6)	There is no compatible data type in ALTIBASE HDB for SQL Server NTEXT type, NVARCHAR is used with maximum precision. It may cause data loss during data migration when actual data precision exceeds maximum NVARCHAR size.

Index	Source	Destination	Issue
22	BINARY	BYTE	
23	IMAGE	BLOB	
24	VARBINARY	BLOB	
25	ALL IDENTITY	NUMERIC(38,0)	
26	UNIQUEIDENTIFIER	VARCHAR(40)	There is no compatible data type in ALTIBASE HDB for SQL Server UNIQUEIDENTIFIER type, so VARCHAR type is used to prevent any data loss.
27	SYSNAME	NVARCHAR(128)	

MySQL to ALTIBASE HDB

Index	Source	Destination	Issue
1	TINYINT	SMALLINT	
2	TINYINT UNSIGNED	SMALLINT	
3	SMALLINT	INTEGER	
4	SMALLINT UNSIGNED	INTEGER	
5	MEDIUMINT	INTEGER	
6	MEDIUMINT UNSIGNED	INTEGER	
7	INT (INTEGER)	INTEGER	Please keep in mind that the minimum value of ALTIBASE INT type (-2,147,483,647) is greater than the minimum value of MySQL INT type (-2,147,483,648).
8	INT UNSIGNED	BIGINT	
9	BIGINT	BIGINT	Please keep in mind that the minimum value of ALTIBASE BIGINTINT type (-9,223,372,036,854,775,807) is greater than the minimum value of MySQL BIGINT type (-9,223,372,036,854,775,808).
10	BIGINT UNSIGNED	NUMERIC(20,0)	There is no compatible data type in ALTIBASE HDB for MySQL BIGINT UNSIGNED type, so NUMERIC type is used to prevent any data loss.

Index	Source	Destination	Issue
11	DECIMAL (NUMERIC)	VARCHAR(70)	There is no compatible data type in ALTIBASE HDB for MySQL DECIMAL type, so VARCHAR type is used to prevent any data loss.
12	FLOAT	FLOAT	
13	DOUBLE	VARCHAR(310)	There is no compatible data type in ALTIBASE HDB for MySQL DOUBLE type, so VARCHAR type is used to prevent any data loss.
14	BIT	VARBIT	
15	DATETIME	DATE	Time parts are set to '0'
16	TIMESTAMP	DATE	Except TIMEZONE
17	CHAR	CHAR	
18	VARCHAR	VARCHAR	
19	CHAR with National Character	NCHAR	
20	VARCHAR with National Character	NVARCHAR	
21	BINARY	BYTE	
22	VARBINARY	BLOB	
23	TINYBLOB	BLOB	
24	MEDIUMBLOB	BLOB	
25	BLOB	BLOB	
26	LONGBLOB	BLOB	
27	TINYTEXT	VARCHAR(255)	
28	TEXT	CLOB	
29	MEDIUMTEXT	CLOB	
30	LONGTEXT	CLOB	
31	ENUM	VARCHAR(10666)	There is no compatible data type in ALTIBASE HDB for MySQL ENUM type, so VARCHAR is used to prevent data loss.
32	SET	VARCHAR(10666)	There is no compatible data type in ALTIBASE HDB for MySQL SET type, so VARCHAR is used to prevent data loss.

Informix 11.5 to ALTIBASE HDB

Index	Source	Destination	Issue
1	BIGINT	BIGINT	
2	INT8	BIGINT	
3	INT	INTEGER	
4	SMALLINT	SMALLINT	
5	BIGSERIAL	BIGINT	
6	SERIAL8	BIGINT	
7	SERIAL	INTEGER	
8	FLOAT	DOUBLE	
9	REAL	REAL	
10	SMALLFLOAT	REAL	
11	MONEY	NUMERIC	
12	DECIMAL_FLOAT	FLOAT	
13	DATE	DATE	
14	DATETIME	DATE	
15	BOOLEAN	CHAR(1)	
16	CHAR	CHAR	The user should keep in mind that data loss can occur due to the maximum precision of CHAR data type at Informix (32,767) being greater than that of ALTIBASE(32,000).
17	NCHAR	NCHAR	The user should keep in mind that data loss can occur due to the maximum precision of NCHAR data type at Informix (32,767) being greater than that of ALTIBASE(32,000).
18	VARCHAR	VARCHAR	
19	NVARCHAR	NVARCHAR	
20	LVARCHAR	VARCHAR	The user should keep in mind that data loss can occur due to the maximum precision of LVARCHAR data type at Informix (32,767) being greater than that of ALTIBASE(32,000).
21	TEXT	CLOB	
22	CLOB	CLOB	The user should keep in mind that data loss can occur due to the maximum size of CLOB data type at Informix (4GB) being greater than that of ALTIBASE(2GB).

Index	Source	Destination	Issue
23	BYTE	BLOB	
24	BLOB	BLOB	The user should keep in mind that data loss can occur due to the maximum size of BLOB data type at Informix (4GB) being greater than that of ALTIBASE(2GB).
25	INTERVAL	FLOAT	

Appendix D. PSM Converter Rule List

Migration Center provides DDL SQL texts for creating PSM type database objects with PSM Converter for Oracle to Altibase Migration.

PSM converter uses rules to convert DDL SQL texts, and these rules fall into three categories as desrcribed below:

- Converted: Convertible
- Removed: Not convertible, but may be removable
- To-Do: Not convertible nor removable

If To-Do rule is applied to a PSM object, then it will show up in To-Do List pane. If not, then it will show up in Done List pane.

On version scope omission, rules apply to all versions; on version scope specification, rules apply to the given version.

Please refer to following list for more detailed information about the rules.

View Conversion Rules

RULE-11001

- Type: REMOVED
- Description: 'WITH CHECK OPTION' is removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule11001testview1
AS
SELECT * FROM t1
WITH CHECK OPTION CONSTRAINT constraint_t;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule11001testview1
AS
SELECT * FROM t1
/* WITH CHECK OPTION CONSTRAINT constraint_t */ /* [REMOVED] RULE-11001 :
'WITH CHECK OPTION' is removed */;
```

RULE-11002

- Type: REMOVED
- Description: Parameter constraints are removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule11002testview1
(
    a1 CONSTRAINT constraint_t PRIMARY KEY
)
AS SELECT c1 FROM t1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule11002testview1
(
    a1 /* CONSTRAINT constraint_t PRIMARY KEY */ /* [REMOVED] RULE-11002
: Parameter constraints are removed */
)
AS SELECT c1 FROM t1;
```

RULE-11003

- Type: TODO
- Description: Out of line constraint must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule11003testview1
(
    CONSTRAINT constraint_t UNIQUE(a1)
)
AS SELECT c1 FROM t1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule11003testview1
(
    CONSTRAINT constraint_t UNIQUE(a1) /* [TODO] RULE-11003 : Out of line
constraint must be converted manually */
)
AS SELECT c1 FROM t1;
```

Trigger Conversion Rules

RULE-12002

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: TODO

- Description: 'INSTEAD OF' must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule12002testtrig1
INSTEAD OF INSERT ON t1
DECLARE
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule12002testtrig1
INSTEAD OF /* [TODO] RULE-12002 : 'INSTEAD OF' must be converted manually
*/ INSERT ON t1
DECLARE
BEGIN
NULL;
END;
```

RULE-12003

- Type: TODO
- Description: Triggers supporting multiple events must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule12003testtrig1
BEFORE INSERT OR DELETE OR UPDATE OF c1, c2, c3 ON t1
DECLARE
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule12003testtrig1
BEFORE INSERT OR DELETE OR UPDATE OF c1, c2, c3 ON t1 /* [TODO] RULE-
12003 : Triggers supporting multiple events must be converted manually */
DECLARE
BEGIN
NULL;
END;
```

RULE-12004

- Type: TODO
- Description: 'AS' or 'IS' must replace 'DECLARE' before PSM body, or be added on omission of 'DECLARE'
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule12004testtrig1
BEFORE INSERT ON t1
DECLARE
```

```
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule12004testtrig2
BEFORE INSERT ON t1
DECLARE /* [TODO] RULE-12004 : 'AS' or 'IS' must replace 'DECLARE' before
PSM body, or be added on omission of 'DECLARE' */
BEGIN
NULL;
END;
```

RULE-12005

- Type: TODO
- Description: Non DML trigger must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule12005testtrig1
BEFORE CREATE ON DATABASE
DECLARE
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule12005testtrig1
BEFORE CREATE ON DATABASE /* [TODO] RULE-12005 : Non DML trigger must be
converted manually */
DECLARE
BEGIN
NULL;
END;
```

RULE-12006

- Type: TODO
- Description: Compound DML trigger must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule12006testtrig1
FOR DELETE ON t1
DECLARE
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule12006testtrig1
FOR DELETE ON t1 /* [TODO] RULE-12006 : Compound DML trigger must be con-
```

```
verted manually */
DECLARE
BEGIN
NULL;
END;
```

RULE-12007

- Type: TODO
- Description: Nested table must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule12007testtrig1
AFTER DELETE ON NESTED TABLE t1 OF v1
DECLARE
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule12007testtrig1
AFTER DELETE ON NESTED TABLE t1 OF v1 /* [TODO] RULE-12007 : Nested table
must be converted manually */
DECLARE
BEGIN
NULL;
END;
```

RULE-12008

- Type: TODO
- Description: CALL routine clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule12008testtrig1
FOR DELETE ON t1
CALL testproc1(a1, a2);
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule12008testtrig1
FOR DELETE ON t1
CALL testproc1(a1, a2); /* [TODO] RULE-12008 : CALL routine clause must
be converted manually */
```

Function Conversion Rules

RULE-13001

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: TODO
- Description: 'AS LANGUAGE' clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule13001testfunc1( a1 IN NUMBER )
RETURN NUMBER
AS LANGUAGE JAVA
NAME 'test.quote() return java.lang.String';
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule13001testfunc1( a1 IN NUMBER )
RETURN NUMBER
AS LANGUAGE JAVA
NAME 'test.quote() return java.lang.String' /* [TODO] RULE-13001 : 'AS
LANGUAGE' clause must be converted manually */;
```

RULE-13002

- Type: REMOVED
- Description: AUTHID clause is removed
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule13002testfunc1( a1 IN NUMBER )
RETURN NUMBER
AUTHID CURRENT_USER
IS
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule13002testfunc1( a1 IN NUMBER )
RETURN NUMBER
/* AUTHID CURRENT_USER */ /* [REMOVED] RULE-13002 : AUTHID clause is
removed */
IS
BEGIN
RETURN a1;
```

RULE-13003

- Type: REMOVED

- Description: PARALLEL_ENABLE clause is removed
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule13003testfunc1( a1 IN NUMBER )
RETURN NUMBER
PARALLEL_ENABLE
IS
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule13003testfunc1( a1 IN NUMBER )
RETURN NUMBER
/* PARALLEL_ENABLE */ /* [REMOVED] RULE-13003 : PARALLEL_ENABLE clause is
removed */
IS
BEGIN
RETURN a1;
END;
```

RULE-13004

- Type: REMOVED
- Description: RESULT_CACHE clause is removed
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule13004testfunc1( a1 IN NUMBER )
RETURN NUMBER
RESULT_CACHE RELIES_ON(t1, t2)
IS
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule13004testfunc2( a1 IN NUMBER )
RETURN NUMBER
/* RESULT_CACHE RELIES_ON(t1, t2) */ /* [REMOVED] RULE-13004 :
RESULT_CACHE clause is removed */
IS
BEGIN
RETURN a1;
END;
```

RULE-13005

- Type: REMOVED
- Description: 'DETERMINISTIC' is removed
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule13005testfunc1( a1 IN NUMBER )
RETURN NUMBER
DETERMINISTIC
IS
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule13005testfunc1( a1 IN NUMBER )
RETURN NUMBER
/* DETERMINISTIC */ /* [REMOVED] RULE-13005 : 'DETERMINISTIC' is removed
*/
IS
BEGIN
RETURN a1;
END;
```

RULE-13006

- Type: REMOVED
- Description: 'PIPELINED' is removed
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule13006testfunc1( a1 IN NUMBER )
RETURN NUMBER
PIPELINED AS
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule13006testfunc1( a1 IN NUMBER )
RETURN NUMBER
/* PIPELINED */ /* [REMOVED] RULE-13006 : 'PIPELINED' is removed */ AS
BEGIN
RETURN a1;
END;
```

RULE-13007

- Type: TODO
- Description: PIPELINED/AGGREGATE USING clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule13007testfunc1( a1 IN NUMBER )
RETURN NUMBER
AGGREGATE USING implementation_type;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule13007testfunc1( a1 IN NUMBER )
```

```
RETURN NUMBER
AGGREGATE USING implementation_type /* [TODO] RULE-13007 : PIPELINED/
AGGREGATE USING clause must be converted manually */;
```

Procedure Conversion Rules

RULE-14001

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: TODO
- Description: 'AS LANGUAGE' clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule14001testproc1( a1 IN NUMBER )
AS LANGUAGE JAVA
NAME 'test.quote() return java.lang.String';
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule14001testproc1( a1 IN NUMBER )
AS LANGUAGE JAVA
NAME 'test.quote() return java.lang.String' /* [TODO] RULE-14001 : 'AS
LANGUAGE' clause must be converted manually */;
```

RULE-14002

- Type: REMOVED
- Description: AUTHID clause is removed
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule14002testproc1( a1 IN NUMBER )
AUTHID DEFINER
IS
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule14002testproc1( a1 IN NUMBER )
/* AUTHID DEFINER */ /* [REMOVED] RULE-14002 : AUTHID clause is removed
*/
IS
BEGIN
NULL;
END;
```

Materialized View Conversion Rules

RULE-15004

- Type: REMOVED
- Description: All clauses between column alias clause and subquery are removed
- Original SQL Text:

```
CREATE MATERIALIZED VIEW rule15004testmview1
ORGANIZATION HEAP PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE test
BUILD IMMEDIATE
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE tbs1
REFRESH FAST ON DEMAND
WITH PRIMARY KEY USING DEFAULT LOCAL ROLLBACK SEGMENT
USING ENFORCED CONSTRAINTS FOR UPDATE DISABLE QUERY REWRITE
AS SELECT * FROM t1;
```

- Processed SQL Text:

```
CREATE MATERIALIZED VIEW rule15004testmview1
/* ORGANIZATION HEAP PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE test
BUILD IMMEDIATE
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE tbs1
REFRESH FAST ON DEMAND
WITH PRIMARY KEY USING DEFAULT LOCAL ROLLBACK SEGMENT
USING ENFORCED CONSTRAINTS FOR UPDATE DISABLE QUERY REWRITE */ /*
[REMOVED] RULE-15004 :  All clauses between column alias clause and sub-
query are removed */
AS SELECT * FROM t1;
```

Package Conversion Rules

RULE-16001

- Type: CONVERTED
- Description: Rule is applied to sub-program in package object
- Original SQL Text:

```
CREATE OR REPLACE PACKAGE BODY rule16001testpkg1
AS
PROCEDURE testproc1
AS
m_rowid ROWID;
BEGIN
DBMS_OUTPUT.ENABLE;
DBMS_OUTPUT.PUT('--- test procedure is executed ---');
DBMS_OUTPUT.DISABLE;
END;
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PACKAGE BODY rule16001testpkg1
AS
PROCEDURE testproc1
AS
m_rowid ROWID /* [TODO] RULE-30002 : Unsupported data type must be con-
verted manually */;
BEGIN
/* DBMS_OUTPUT.ENABLE; */ /* [REMOVED] RULE-32005 : Unsupported built-in
package sub-procedure is removed */
SYSTEM_.PRINT('--- test procedure is executed ---');
/* DBMS_OUTPUT.DISABLE; */ /* [REMOVED] RULE-32005 : Unsupported built-in
package sub-procedure is removed */
END;
BEGIN
NULL;
END;
```

Library Conversion Rules

RULE-17001

- Type: REMOVED
- Description: Agent clause is removed
- Original SQL Text:

```
CREATE OR REPLACE LIBRARY rule17001testlib1 AS
'${ORACLE_HOME}/lib/test_lib.so' AGENT 'test.rule.no_17001.com';
```

- Processed SQL Text:

```
CREATE OR REPLACE LIBRARY rule17001testlib1 AS
'${ORACLE_HOME}/lib/test_lib.so' /* AGENT 'test.rule.no_17001.com' */ /*
[REMOVED] RULE-17001 : Agent clause is removed */;
```

RULE-17002

- Type: REMOVED

- Description: 'UNTRUSTED' is removed
- Original SQL Text:

```
CREATE OR REPLACE LIBRARY rule17001testlib1 UNTRUSTED
AS '${ORACLE_HOME}/lib/test_lib.so';
```

- Processed SQL Text:

```
CREATE OR REPLACE LIBRARY rule17001testlib1  /* UNTRUSTED */ /* [REMOVED]
RULE-17002 : 'UNTRUSTED' is removed */
AS '${ORACLE_HOME}/lib/test_lib.so';;
```

DML Conversion Rules

RULE-20001

- Type: REMOVED
- Description: Flashback query clause is removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20001testview1
AS
SELECT * FROM t1 CROSS JOIN t2 VERSIONS BETWEEN TIMESTAMP MINVALUE AND
MAXVALUE;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20001testview1
AS
SELECT * FROM t1 CROSS JOIN t2 /* VERSIONS BETWEEN TIMESTAMP MINVALUE AND
MAXVALUE */ /* [REMOVED] RULE-20001 : Flashback query clause is removed
*/;
```

RULE-20002

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: CONVERTED
- Description: ROLLUP operation in the GROUP BY clause is converted
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20002testview1
AS
SELECT c1, c2, c3, c4, SUM( c5 )
FROM t1
WHERE c1 > c2
GROUP BY ROLLUP( c1, c2 , ( c3, c4 ) );
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20002testview1
AS
SELECT c1, c2, c3, c4, SUM( c5 )
FROM (
SELECT DECODE(no, 1, c1, 2, c1, 3, c1) c1, DECODE(no, 1, c2, 2, c2) c2,
DECODE(no, 1, c3) c3, DECODE(no, 1, c4) c4, c5
FROM t1, (SELECT LEVEL no FROM dual CONNECT BY LEVEL <= 4) copy_t
WHERE c1 > c2
)
GROUP BY c1, c2, c3, c4;
```

RULE-20003

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: CONVERTED
- Description: CUBE operation in the GROUP BY clause is converted
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20003testview1
AS
SELECT c1,
  c2,
  c3,
  c4,
  TO_CHAR(SUM( c5 ), '9,999,999,999'),
  RANK() OVER (PARTITION BY GROUPING_ID(c1, c2, c3, c4) ORDER BY SUM(c5)
DESC)
FROM t1
GROUP BY CUBE( c1, c2 ), c3, c4;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20003testview1
AS
SELECT NVL(c1, '') AS "C1",
   NVL(c2, '') AS "C2",
   NVL(c3, '') AS "C3",
   NVL(c4, '') AS "C4",
   TO_CHAR(SUM( c5 ), '9,999,999,999'),
   RANK() OVER (PARTITION BY GROUPING_ID(c1, c2, c3, c4) ORDER BY SUM(c5)
DESC)
FROM (
     SELECT DECODE(no, 1, c1, 3, c1) c1, DECODE(no, 1, c2, 4, c2) c2, c3,
c4, c5, no
      FROM t1, (SELECT LEVEL no FROM dual CONNECT BY LEVEL <= 4) copy_t
)
GROUP BY c1, c2, c3, c4;
```

RULE-20004

- Type: TODO
- Description: Unsupported Function
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule20004testproc1( a1 NUMBER )
AS BEGIN
INSERT INTO t1
VALUES (
  First_value(a1) OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING),
  Last_value(a1) OVER (ORDER BY salary ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING)
  );
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule20004testproc1( a1 NUMBER )
AS BEGIN
INSERT INTO t1
VALUES (
  First_value(a1) OVER (ORDER BY salary ASC ROWS UNBOUNDED PRECEDING) /*
[TODO] RULE-20004 : Unsupported Function */,
  Last_value(a1) OVER (ORDER BY salary ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) /* [TODO] RULE-20004 : Unsupported Function */,
  );
END;
```

RULE-20005

- Type: TODO
- Description: OVER clause must be reviewed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20005testview1
AS
SELECT c1,
  c2,
  c3,
  c4,
  SUM( c5 ) OVER (PARTITION BY c1_1 ORDER BY c1_1 RANGE UNBOUNDED PRECED-
ING)
FROM t1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20005testview1
AS
SELECT c1,
  c2,
  c3,
  c4,
  SUM( c5 ) OVER (PARTITION BY c1_1 ORDER BY c1_1 RANGE UNBOUNDED PRECED-
ING) /* [TODO] RULE-20005 : OVER clause must be reviewed */
FROM t1;
```

RULE-20006

- Type: TODO
- Description: DBlink must be converted manually

- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20006testview1
AS
SELECT c1, c2, c3, c4, SUM( c5 ) FROM t1@remote;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20006testview1
AS
SELECT c1, c2, c3, c4, SUM( c5 ) FROM t1@remote /* [TODO] RULE-20006 :
DBlink must be converted manually */;
```

RULE-20007

- Type: TODO
- Description: GROUPING SETS clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20007testview1
AS
SELECT c1, c2, c3, c4, SUM( c5 )
FROM t1
GROUP BY GROUPING SETS( (c1, c2, c3, c4), (c1, c2, c3), (c3, c4) );
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20007testview1
AS
SELECT c1, c2, c3, c4, SUM( c5 )
FROM t1
GROUP BY GROUPING SETS( (c1, c2, c3, c4), (c1, c2, c3), (c3, c4) ) /*
[TODO] RULE-20007 : GROUPING SETS clause must be converted manually */;
```

RULE-20008

- Type: TODO
- Description: CONNECT BY clause with multiple conditions must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20008testview1
AS
SELECT c1, c2, c3, c4 FROM t1 CONNECT BY c1 = c2 AND c3 > c4 START WITH
c1 = c4;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20008testview1
AS
SELECT c1, c2, c3, c4 FROM t1 CONNECT BY c1 = c2 AND c3 > c4 /* [TODO]
RULE-20008 : CONNECT BY clause having multiple conditions must be con-
verted manually */ START WITH c1 = c4;
```

RULE-20009

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: TODO
- Description: START WITH clause after CONNECT BY clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20009testview1
AS
SELECT c1, c2, c3, c4 FROM t1 CONNECT BY c1 = c2 START WITH c1 = c4;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20009testview1
AS
SELECT c1, c2, c3, c4 FROM t1 CONNECT BY c1 = c2 START WITH c1 = c4 /*
[TODO] RULE-20009 : START WITH clause after CONNECT BY clause must be
converted manually */;
```

RULE-20010

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: TODO
- Description: To convert 'NOCYCLE', 'IGNORE LOOP' should come after condition
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20010testview1
AS
SELECT c1, c2, c3, c4
FROM t1 CONNECT BY NOCYCLE c1 = c2 START WITH c1 = c4;
Processed SQL Text:
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20010testview1
AS
SELECT c1, c2, c3, c4
FROM t1 CONNECT BY NOCYCLE /* [TODO] RULE-20010 : To convert 'NOCYCLE',
'IGNORE LOOP' should come after condition */ c1 = c2 START WITH c1 = c4;
```

RULE-20011

- Type: REMOVED
- Description: All hints are removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20011testview1
AS
SELECT /*+ORDERED */ c1, c2, c3, c4 FROM t1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20011testview1
AS
SELECT c1, c2, c3, c4 FROM t1;
```

RULE-20012

- Version Scope: Versions prior to ALTIBASE HDB 6.1.1.0.0
- Type: TODO
- Description: PIVOT clause must be reviewed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20012testview1
AS
SELECT *
FROM (SELECT c1, c2, c3 FROM pivot_test)
PIVOT(SUM(c4) AS args_sum FOR c1_1 IN ('A' AS a, 'B' AS b, 'C' AS c, 'D'
AS d))
ORDER BY c1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20012testview1
AS
SELECT *
FROM (SELECT c1, c2, c3 FROM pivot_test)
PIVOT(SUM(c4) AS args_sum FOR c1_1 IN ('A' AS a, 'B' AS b, 'C' AS c, 'D'
AS d)) /* [TODO] RULE-20012 : PIVOT clause must be reviewed */
ORDER BY c1;
```

RULE-20013

- Type: TODO
- Description: UNPIVOT clause must be reviewed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20013testview1
AS
SELECT * FROM t1
UNPIVOT (c5 FOR c2 IN (c3 AS 'no', c4 AS 'name'))
ORDER BY c1, c2;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20013testview1
AS
SELECT * FROM t1
UNPIVOT (c5 FOR c2 IN (c3 AS 'no', c4 AS 'name')) /* [TODO] RULE-20013 :
UNPIVOT clause must be reviewed */
ORDER BY c1, c2;
```

RULE-20014

- Type: CONVERTED
- Description: Schema name is removed
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE testuser1.rule20014testproc1(a1 IN NUMBER)
AS
BEGIN
INSERT INTO testuser1.t1 VALUES(1, 2, 3);
UPDATE testuser2.t1 SET c1 =3, c2 = c2 + 4, c3 = 9 WHERE c4 = 12;
DELETE FROM TESTUSER1.t1 WHERE c4 = 12;
SELECT * INTO :cur1, :cur2 FROM "TEST_USER1".t1;
SELECT * INTO :cur1, :cur2 FROM "Test_User1".t1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule20014testproc1(a1 IN NUMBER)
AS
BEGIN
INSERT INTO t1 VALUES(1, 2, 3);
UPDATE testuser2.t1 SET c1 =3, c2 = c2 + 4, c3 = 9 WHERE c4 = 12;
DELETE FROM t1 WHERE C4 = 12;
SELECT * INTO :cur1, :cur2 FROM t1;
SELECT * INTO :cur1, :cur2 FROM "Test_User1".t1;
END;
```

RULE-20015

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: CONVERTED
- Description: RETURNING clause is converted
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule20015testproc(a1 IN NUMBER)
AS
BEGIN
INSERT INTO t1 VALUES(1, 2, 3) RETURNING c1, c2, c3 INTO :bnd1, :bnd2,
:bnd3;
UPDATE t1 SET c1 =3, c2 = c2 + 4, c3 = 9 WHERE c4 = 12 RETURNING c1, c2,
c3 INTO :bnd1, :bnd2, :bnd3;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule20015testproc(a1 IN NUMBER)
AS
BEGIN
INSERT INTO t1 VALUES(1, 2, 3);
SELECT c1, c2, c3 INTO :bnd1, :bnd2, :bnd3 FROM t1;
UPDATE t1 SET c1 =3, c2 = c2 + 4, c3 = 9 WHERE c4 = 12;
SELECT c1, c2, c3 INTO :bnd1, :bnd2, :bnd3 FROM t1 WHERE c4 = 12;
END;
```

RULE-20016

- Type: TODO
- Description: CONNECT_BY_ISCYCLE clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20016testview1
AS
SELECT c1,
  CONNECT_BY_ISCYCLE "IsCycle",
  LEVEL,
  SYS_CONNECT_BY_PATH(c1, '/') "Path"
FROM t1
WHERE LEVEL <= 3
START WITH c2 = 100
CONNECT BY PRIOR c2 = c3 AND LEVEL <= 4;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20016testview1
AS
SELECT c1,
  CONNECT_BY_ISCYCLE "IsCycle" /* [TODO] RULE-20016 : CONNECT_BY_ISCYCLE
clause must be converted manually */,
  LEVEL,
  SYS_CONNECT_BY_PATH(c1, '/') "Path"
FROM t1
WHERE LEVEL <= 3
START WITH c2 = 100
CONNECT BY PRIOR c2 = c3 AND LEVEL <= 4;
```

RULE-20017

- Version Scope: Versions prior to ALTIBASE HDB 6.1.1.0.0
- Type: REMOVED
- Description: 'NULLS FIRST' and 'NULLS LAST' are removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20017testview1
AS
SELECT * FROM t1
ORDER BY c1,
  c2 ASC NULLS FIRST,
  c3 NULLS LAST;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20017testview1
AS
SELECT * FROM t1
ORDER BY c1,
  c2 ASC /* NULLS FIRST */ /* [REMOVED] RULE-20017 : 'NULLS FIRST' and
'NULLS LAST' are removed */,
  c3 /* NULLS LAST */ /* [REMOVED] RULE-20017 : 'NULLS FIRST' and 'NULLS
LAST' are removed */;
```

- Version Scope: ALTIBASE HDB 6.1.1.0.0 and above
- Type: REMOVED
- Description: 'NULLS FIRST' and 'NULLS LAST' are removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20017testview1
AS
SELECT c1,
  c2,
  c3,
  c4,
  RANK() OVER (PARTITION BY c1 ORDER BY c3 DESC, c4 NULLS LAST)
FROM t1
ORDER BY c1, c2, c3, c4  NULLS LAST;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20017testview1
AS
SELECT c1,
  c2,
  c3,
  c4,
  RANK() OVER (PARTITION BY c1 ORDER BY c3 DESC, c4 /* NULLS LAST */ /*
[REMOVED] RULE-20017 : 'NULLS FIRST' and 'NULLS LAST' are removed */)
FROM t1
ORDER BY c1, c2, c3, c4  /* NULLS LAST */ /* [REMOVED] RULE-20017 :
'NULLS FIRST' and 'NULLS LAST' are removed */;
```

RULE-20018

- Type: TODO
- Description: ONLY clause must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20018testview1
AS
SELECT c1, c2 FROM ONLY(testview1);
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20018testview1
AS
SELECT c1, c2 FROM ONLY(testview1) /* [TODO] RULE-20018 : ONLY clause
must be converted manually */;
```

RULE-20019

- Type: REMOVED
- Description: Restriction clause is removed

- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20019testview1
AS
SELECT c1, c2 FROM (SELECT c1, c2 FROM t2 WITH READ ONLY) t1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20019testview1
AS
SELECT c1, c2 FROM (SELECT c1, c2 FROM t2 /* WITH READ ONLY */ /*
[REMOVED] RULE-20019 : Restriction clause is removed */) t1;
```

RULE-20020

- Type: TODO
- Description: Inner join clause that is a CROSS or NATURAL join must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20020testview1
AS
SELECT c1, c2 FROM (SELECT * FROM testview1) CROSS JOIN testview2;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20020testview1
AS
SELECT c1, c2 FROM (SELECT * FROM testview1) CROSS JOIN testview2 /*
[TODO] RULE-20020 : Inner join clause that is a CROSS or NATURAL join
must be converted manually */;
```

RULE-20021

- Type: TODO
- Description: USING clause in a join must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20017testview1
AS
SELECT c1, c2 FROM t1 JOIN testview1 USING(c1, c2);
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20017testview1
AS
SELECT c1, c2 FROM t1 JOIN testview1 USING(c1, c2) /* [TODO] RULE-20021 :
USING clause in a join must be converted manually */;
```

RULE-20023

- Type: REMOVED

- Description: 'UNIQUE' keyword is removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20023testview1
AS
SELECT UNIQUE c1 FROM t1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20023testview1
AS
SELECT /* UNIQUE */ /* [REMOVED] RULE-20023 : 'UNIQUE' keyword is removed
*/ c1 FROM t1;
```

RULE-20024

- Type: TODO
- Description: 'LEADING', 'TRAILING', 'BOTH', and 'FROM' keywords in TRIM Function must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20024testview1
AS
SELECT c1, TO_CHAR(TRIM(TRAILING 0 FROM c3))
FROM t1 WHERE c2 = 60 ORDER BY c1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20024testview1
AS
SELECT c1, TO_CHAR(TRIM(TRAILING 0 FROM c3) /* [TODO] RULE-20024 :
'LEADING', 'TRAILING', 'BOTH', and 'FROM' keywords in TRIM Function must
be converted manually */)
FROM t1 WHERE c2 = 60 ORDER BY c1;
```

RULE-20025

- Type: TODO
- Description: BIN_TO_NUM function taking multiple arguments must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20024testview1
AS
SELECT BIN_TO_NUM(1,0,1,0) FROM DUAL;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20024testview1
AS
SELECT BIN_TO_NUM(1,0,1,0) /* [TODO] RULE-20025 : BIN_TO_NUM function
taking multiple arguments must be converted manually */ FROM DUAL;
```

RULE-20026

- Type: TODO
- Description: CAST function taking a subquery as an argument must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20026testview1
AS
SELECT c1,CAST(MULTISET(SELECT c1
  FROM t2
  ORDER BY c2) AS tmp_tbl
  )
FROM t1 ORDER BY c1;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20026testview1
AS
SELECT c1,CAST(MULTISET(SELECT c1
  FROM t2
  ORDER BY c2) AS tmp_tbl
  ) /* [TODO] RULE-20026 : CAST function taking a subquery as an argument
must be converted manually */
FROM t1 ORDER BY c1;
```

RULE-20027

- Type: TODO
- Description: DUMP function taking multiple arguments must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE VIEW rule20027testview1
AS
SELECT DUMP(c3, 8, 3, 2) FROM t1 WHERE c3 = 100 ORDER BY c2;
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW rule20027testview1
AS
SELECT DUMP(c3, 8, 3, 2) /* [TODO] RULE-20025 :  DUMP function taking
multiple arguments must be converted manually */ FROM t1 WHERE c3 = 100
ORDER BY c2;
```

RULE-20028

- Type: CONVERTED
- Description: Double quotations are removed
- Original SQL Text:

```
CREATE OR REPLACE VIEW "TESTUSER1"."RULE20028TESTVIEW1" ( "A1", "A2",
"A3", "A4", "A5" )
```

```
AS
SELECT "CODE" "A1", "name" "A2", '1' "A3", '??o' "A4", '??o' "A5" FROM
"T1"
UNION ALL
SELECT code A1, name A2, '2' A3, '?6??' A4, '?6??' A5 FROM T2
UNION ALL
SELECT "no" "A1", "nm" "A2", '3' "A3", '????' "A4", '????' "A5" FROM "T3"
WHERE "C6" = '2';
```

- Processed SQL Text:

```
CREATE OR REPLACE VIEW TESTUSER1.RULE20028TESTVIEW1 ( A1, A2, A3, A4, A5
)
AS
SELECT CODE A1, name A2, '1' A3, '??o' A4, '??o' A5 FROM T1
UNION ALL
SELECT code A1, name A2, '2' A3, '?6??' A4, '?6??' A5 FROM T2
UNION ALL
SELECT no A1, nm A2, '3' A3, '????' A4, '????' A5 FROM T3 WHERE C6 = '2';
```

RULE-20029

- Type: CONVERTED
- Description: Name of identifier that is an ALTIBASE keyword is converted
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE wakeup_recptr
AS
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE wakeup_recptr_POC
AS
BEGIN
NULL;
END;
```

PSM Conversion Rules

RULE-30001

- Type: CONVERTED
- Description: Unsupported data type is converted
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule30001testfunc1( a1 NUMBER )
RETURN NUMBER
IS
m_binary_double BINARY_DOUBLE;
```

```
m_number NUMBER(10)  := a1;
TYPE typ6 IS RECORD ( m1 NATURAL );
TYPE typ7 IS RECORD ( m1 NATURALN );
TYPE typ35 IS TABLE OF TIMESTAMP(3) INDEX BY VARCHAR2(10);
TYPE typ36 IS TABLE OF TIMESTAMP(3) WITH TIME ZONE INDEX BY VARCHAR2(10);
BEGIN
RETURN m_number;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule30001testfunc1( a1 NUMBER(38, 0) )
RETURN NUMBER(38, 0)
IS
m_binary_double VARCHAR(310);
m_number NUMBER(38, 0) := a1;
TYPE typ6 IS RECORD ( m1 INTEGER );
TYPE typ7 IS RECORD ( m1 INTEGER );
TYPE typ35 IS TABLE OF DATE INDEX BY VARCHAR2(10);
TYPE typ36 IS TABLE OF DATE INDEX BY VARCHAR2(10);
BEGIN
RETURN m_number;
END;
```

RULE-30002

- Type: TODO
- Description: Unsupported data type must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule30002testproc1
IS
m_rowid ROWID;
m_urowid UROWID;
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule30002testproc1
IS
m_rowid ROWID /* [TODO] RULE-30002 : Unsupported data type must be con-
verted manually */;
m_urowid UROWID /* [TODO] RULE-30002 : Unsupported data type must be con-
verted manually */;
BEGIN
NULL;
END;
```

RULE-30003

- Type: TODO
- Description: All data types of collection name, cursor variable name, or object name with %TYPE must be converted manually

- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule30003testproc1
(
    a1 IN NUMBER,
    a2 collection_name%TYPE
)
IS
tmp1 typ1%TYPE;
CURSOR cur1
(
    m1 collection_name%TYPE,
    m2 collection_name%TYPE
) RETURN t2%ROWTYPE
IS SELECT c2, c3 FROM t1 WHERE c1 > 10;
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule30003testproc1
(
    a1 IN NUMBER,
    a2 collection_name%TYPE /* [TODO] RULE-30003 : All data types of col-
lection name, cursor variable name, or object name with %TYPE must be
converted manually */
)
IS
tmp1 typ1%TYPE /* [TODO] RULE-30003 : All data types of collection name,
cursor variable name or object name with %TYPE must be converted manually
*/;
CURSOR cur1
(
    m1 collection_name%TYPE /* [TODO] RULE-30003 :  All data types of
collection name, cursor variable name or object name with %TYPE must be
converted manually */,
    m2 collection_name%TYPE /* [TODO] RULE-30003 :  All data types of
collection name, cursor variable name or object name with %TYPE must be
converted manually */
) RETURN t2%ROWTYPE
IS SELECT c2, c3 FROM t1 WHERE c1 > 10;
BEGIN
NULL;
END;
```

RULE-30004

- Type: TODO
- Description: Collection, object, record, or REF cursor type variables must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule30004testtrig1
BEFORE DELETE ON t1
DECLARE
m1 NUMBER;
m2 REF object_name;
CURSOR cur1
```

```
(
    a1 collection_name%TYPE,
    a2 collection_name%TYPE,
    a3 t1.c1%ROWTYPE
) RETURN t2%ROWTYPE
IS SELECT c1, c2, c3 FROM t1 WHERE c4 > 2000;
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule30004testtrig1
BEFORE DELETE ON t1
DECLARE
m1 NUMBER;
m2 REF object_name /* [TODO] RULE-30004 :Data types that are type names
of a collection, object, record, or REF cursor type variable must be con-
verted manually */;
CURSOR cur1
(
    a1 collection_name%TYPE,
    a2 collection_name%TYPE,
    a3 t1.c1%ROWTYPE /* [TODO] RULE-30004 : Data types that are type
names of a collection, object, record, or REF cursor type variable must
be converted manually */
) RETURN t2%ROWTYPE
IS SELECT c1, c2, c3 FROM t1 WHERE c4 > 2000;
BEGIN
NULL;
END;
```

RULE-30005

- Type: TODO
- Description: 'NOT NULL' in DECLARE section must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule30005testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
v1 PLS_INTEGER NOT NULL;
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule30005testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
v1 PLS_INTEGER NOT NULL /* [TODO] RULE-30005 : 'NOT NULL' in DECLARE sec-
tion must be converted manually */;
BEGIN
RETURN a1;
END;
```

RULE-30006

- Type: CONVERTED
- Description: 'NOCOPY' must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule30006testproc1
(
    a1 OUT NOCOPY NUMBER
)
IS
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule30006testproc1
(
    a1 OUT NOCOPY NUMBER /* [TODO] RULE-30006 : 'NOCOPY' must be con-
verted manually */
)
IS
BEGIN
NULL;
END;
```

RULE-31001

- Type: CONVERTED
- Description: All implicit cursors are converted to explicit cursors
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule31001testproc1( a1 IN NUMBER )
IS
BEGIN
FOR item1 IN (SELECT c1 FROM t1)
LOOP
NULL;
END LOOP;
FOR item2 IN (SELECT c1 FROM t2)
LOOP
NULL;
END LOOP;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule31001testproc1( a1 IN NUMBER )
IS
CURSOR O2A_generated_cur_00 IS (SELECT c1 FROM t1);
CURSOR O2A_generated_cur_01 IS (SELECT c1 FROM t2);
BEGIN
FOR item1 IN O2A_generated_cur_00
LOOP
NULL;
```

```
END LOOP;
FOR item2 IN O2A_generated_cur_01
LOOP
NULL;
END LOOP;
END;
```

RULE-31002

- Type: TODO
- Description: SUBTYPE type variable must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule31002testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
TYPE typ1 IS RECORD ( m1 NUMBER(4) NOT NULL := 99 );
TYPE typ2 IS REF CURSOR RETURN record_name%TYPE;
TYPE typ3 IS TABLE OF a1%TYPE NOT NULL;
TYPE typ4 iS VARYING ARRAY(10) OF INTEGER;
SUBTYPE subtyp1 IS CHAR(10);
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule31002testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
TYPE typ1 IS RECORD ( m1 NUMBER(4) NOT NULL := 99 );
TYPE typ2 IS REF CURSOR RETURN record_name%TYPE;
TYPE typ3 IS TABLE OF a1%TYPE NOT NULL;
TYPE typ4 iS VARYING ARRAY(10) OF INTEGER;
SUBTYPE subtyp1 IS CHAR(10) /* [TODO] RULE-31002 : SUBTYPE type variable
must be converted manually */;
BEGIN
RETURN a1;
END;
```

RULE-31003

- Type: TODO
- Description: VARRAY type variable must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule31003testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
TYPE typ1 IS RECORD ( m1 NUMBER(4) NOT NULL := 99 );
TYPE typ2 IS REF CURSOR RETURN record_name%TYPE;
TYPE typ3 IS TABLE OF a1%TYPE NOT NULL;
TYPE typ4 iS VARYING ARRAY(10) OF INTEGER;
SUBTYPE subtyp1 IS CHAR(10);
BEGIN
```

```
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule31003testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
TYPE typ1 IS RECORD ( m1 NUMBER(4) NOT NULL := 99 );
TYPE typ2 IS REF CURSOR RETURN record_name%TYPE;
TYPE typ3 IS TABLE OF a1%TYPE NOT NULL;
TYPE typ4 iS VARYING ARRAY(10) OF INTEGER /* [TODO] RULE-31003 : VARRAY
type variable must be converted manually */;
SUBTYPE subtyp1 IS CHAR(10);
BEGIN
RETURN a1;
END;
```

RULE-31004

- Type: TODO
- Description: %ROWTYPE type parameter for CURSOR must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule31004testfunc1( a1 IN NUMBER )
RETURN t2%ROWTYPE
IS
CURSOR cur1
(
    m1 collection_name%TYPE,
    m2 t1.c3%ROWTYPE
) RETURN t2%ROWTYPE
IS SELECT c2, c3 FROM t1 WHERE c1 > 10;
BEGIN
RETURN cur1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule31004testfunc1( a1 IN NUMBER )
RETURN t2%ROWTYPE
IS
CURSOR cur1
(
    m1 collection_name%TYPE,
    m2 t1.c3%ROWTYPE /* [TODO] RULE-31004 : %ROWTYPE type parameter for
CURSOR must be converted manually */
) RETURN t2%ROWTYPE
IS SELECT c2, c3 FROM t1 WHERE c1 > 10;
BEGIN
RETURN cur1;
END;
```

RULE-31005

- Type: TODO

- Description: RETURN clause of CURSOR must be converted manually
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule31005testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
TYPE typ1 IS REF CURSOR RETURN record_name%TYPE;
CURSOR cur1 ( m1 NUMBER )
RETURN NUMBER
IS SELECT c2, c3 FROM t1 WHERE c1 > 10;
BEGIN
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule31005testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
TYPE typ1 IS REF CURSOR RETURN record_name%TYPE /* [TODO] RULE-31005 :
RETURN clause of CURSOR must be converted manually */;
CURSOR cur1 ( m1 NUMBER ) RETURN NUMBER /* [TODO] RULE-31005 : RETURN
clause of CURSOR must be converted manually */
IS SELECT c2, c3 FROM t1 WHERE c1 > 10;
BEGIN
RETURN a1;
END;
```

RULE-31006

- Type: REMOVED
- Description: Cannot define or declare PROCEDURE in DECLARE section
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule31006testproc1( a1 IN NUMBER )
IS
PROCEDURE inproc1 (a1 NUMBER);
PROCEDURE inproc2 (a1 NUMBER)
IS
BEGIN
INSERT INTO t1 VALUES(1, a1);
END;
BEGIN
inproc1(tmp1.m1);
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule31006testproc1( a1 IN NUMBER )
IS
/* PROCEDURE inproc1 (a1 NUMBER); */ /* [REMOVED] RULE-31006 : Cannot
define or declare PROCEDURE in DECLARE section */
/* PROCEDURE inproc2 (a1 NUMBER)
IS
BEGIN
INSERT INTO t1 VALUES(1, a1);
END; */ /* [REMOVED] RULE-31006 : Cannot define or declare PROCEDURE in
DECLARE section */
```

```
BEGIN
inproc1(tmp1.m1);
END;
```

RULE-31007

- Type: REMOVED
- Description: Cannot define or declare FUNCTION in DECLARE section
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule31007testproc1( a1 IN NUMBER )
IS
FUNCTION infunc1(a1 NUMBER) RETURN NUMBER;
FUNCTION infunc2(a1 NUMBER)
RETURN NUMBER
AS
BEGIN
RETURN a1;
END;
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule31007testproc1( a1 IN NUMBER )
IS
/* FUNCTION infunc1(a1 NUMBER) RETURN NUMBER; */ /* [REMOVED] RULE-31007
: Cannot define or declare FUNCTION in DECLARE section */
/* FUNCTION infunc2(a1 NUMBER)
RETURN NUMBER
AS
BEGIN
RETURN a1;
END; */ /* [REMOVED] RULE-31007 : Cannot define or declare FUNCTION in
DECLARE section */
BEGIN
NULL;
END;
```

RULE-31008

- Type: REMOVED
- Description: PRAGMA is removed
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule31008testtrig1
BEFORE DELETE ON t1
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule31008testtrig1
BEFORE DELETE ON t1
DECLARE
/* PRAGMA AUTONOMOUS_TRANSACTION; */ /* [REMOVED] RULE-31008 : PRAGMA is
removed */
BEGIN
NULL;
END;;
```

RULE-31009

- Type: REMOVED
- Description: Unsupported exception
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule31009testtrig1
BEFORE DELETE ON t1
DECLARE
ZERO_DIVIDE EXCEPTION;
BEGIN
RETURN a1;
EXCEPTION
WHEN CASE_NOT_FOUND THEN
INSERT INTO tmp VALUES (NULL, NULL, 'Not found');
WHEN ACCESS_INTO_NULL THEN
DBMS_OUTPUT.PUT_LINE('This employee does not receive a commission.');
commission := 0;
WHEN OTHERS THEN
NULL;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule31009testtrig1
BEFORE DELETE ON t1
DECLARE
/* ZERO_DIVIDE EXCEPTION; */ /* [REMOVED] RULE-31009 : Unsupported excep-
tion */
BEGIN
RETURN a1;
EXCEPTION
/* WHEN CASE_NOT_FOUND THEN
INSERT INTO tmp VALUES (NULL, NULL, 'Not found');
*/ /* [REMOVED] RULE-31009 : Unsupported exception */ /* WHEN
ACCESS_INTO_NULL THEN
DBMS_OUTPUT.PUT_LINE('This employee does not receive a commission.');
commission := 0;
*/ /* [REMOVED] RULE-31009 : Unsupported exception */WHEN OTHERS THEN
NULL;
END;
```

RULE-32001

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: REMOVED

- Description: Cannot COMMIT while cursor is still open
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule32001testproc1( a1 IN NUMBER )
AS
m1 INTEGER;
m2 INTEGER;
m3 INTEGER;
m4 INTEGER;
CURSOR cur1 IS
SELECT c1, c2, c3, c4 FROM t1;
BEGIN
OPEN cur1;
FOR i IN 1 .. 5 LOOP
FETCH cur1 INTO m1, m2, m3, m4;
EXIT WHEN cur1%NOTFOUND;
INSERT INTO t2 VALUES(m1, m2, m3, m4);
END LOOP;
COMMIT;
CLOSE cur1;
COMMIT;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule32001testproc1( a1 IN NUMBER )
AS
m1 INTEGER;
m2 INTEGER;
m3 INTEGER;
m4 INTEGER;
CURSOR cur1 IS
SELECT c1, c2, c3, c4 FROM t1;
BEGIN
OPEN cur1;
FOR i IN 1 .. 5 LOOP
FETCH cur1 INTO m1, m2, m3, m4;
EXIT WHEN cur1%NOTFOUND;
INSERT INTO t2 VALUES(m1, m2, m3, m4);
END LOOP;
/* COMMIT; */ /* [REMOVED] RULE-32001 : Cannot COMMIT while cursor is
still open */
CLOSE cur1;
COMMIT;
END;
```

RULE-32002

- Version Scope: Versions prior to ALTIBASE HDB 6.3.1.0.0
- Type: REMOVED
- Description: Unsupported data type is converted
- Original SQL Text:

```
CREATE OR REPLACE TRIGGER rule32002testtrig1
BEFORE DELETE ON t1
DECLARE
m1 INTEGER;
```

```
m2 INTEGER;
m3 INTEGER;
m4 INTEGER;
CURSOR cur1 IS
SELECT c1, c2, c3, c4 FROM t1;
BEGIN
OPEN cur1;
FOR i IN 1 .. 5 LOOP
FETCH cur1 INTO m1, m2, m3, m4;
EXIT WHEN cur1%NOTFOUND;
INSERT INTO t2 VALUES(m1, m2, m3, m4);
END LOOP;
ROLLBACK;
CLOSE cur1;
ROLLBACK;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE TRIGGER rule32002testtrig1
BEFORE DELETE ON t1
DECLARE
m1 INTEGER;
m2 INTEGER;
m3 INTEGER;
m4 INTEGER;
CURSOR cur1 IS
SELECT c1, c2, c3, c4 FROM t1;
BEGIN
OPEN cur1;
FOR i IN 1 .. 5 LOOP
FETCH cur1 INTO m1, m2, m3, m4;
EXIT WHEN cur1%NOTFOUND;
INSERT INTO t2 VALUES(m1, m2, m3, m4);
END LOOP;
/* ROLLBACK; */ /* [REMOVED] RULE-32002 : Cannot ROLLBACK while cursor is
still open */

CLOSE cur1;
ROLLBACK;
END;
```

RULE-32003

- Type: REMOVED
- Description: ‘SET TRANSACTION’ statement is removed
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule32003testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
BEGIN
SET TRANSACTION READ ONLY NAME 'Test Rule 13019';
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule32003testfunc1( a1 IN NUMBER )
```

```
RETURN NUMBER
IS
BEGIN
/* SET TRANSACTION READ ONLY NAME 'Test Rule 13019'; */ /* [REMOVED]
RULE-32003 : 'SET TRANSACTION' statement is removed */
RETURN a1;
END;
```

RULE-32004

- Type: CONVERTED
- Description: Procedures related to printing and file control are converted
- Original SQL Text:

```
CREATE OR REPLACE FUNCTION rule32004testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
BEGIN
DBMS_OUTPUT.PUT('something comments to print or file name');
DBMS_OUTPUT.PUT_LINE('something comments to print or file name');
RETURN a1;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE FUNCTION rule32004testfunc1( a1 IN NUMBER )
RETURN NUMBER
IS
BEGIN
SYSTEM_.PRINT('something comments to print or file name');
SYSTEM_.PRINTLN('something comments to print or file name');
RETURN a1;
END;
```

RULE-32005

- Type: REMOVED
- Description: Unsupported built-in package sub-procedure is removed
- Original SQL Text:

```
CREATE OR REPLACE PROCEDURE rule32005testproc1
IS
line VARCHAR2;
status INTEGER;
lines CHARACTER;
numlines INTEGER;
BEGIN
RAISE TOO_MANY_ROWS;
EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.DISABLE;
DBMS_OUTPUT.ENABLE;
END;
```

- Processed SQL Text:

```
CREATE OR REPLACE PROCEDURE rule32005testproc1
IS
line VARCHAR2;
status INTEGER;
lines CHARACTER;
numlines INTEGER;
BEGIN
RAISE TOO_MANY_ROWS;
EXCEPTION
WHEN TOO_MANY_ROWS THEN
/* DBMS_OUTPUT.DISABLE; */ /* [REMOVED] RULE-32005 : Unsupported built-in
package sub-procedure is removed */
/* DBMS_OUTPUT.ENABLE; */ /* [REMOVED] RULE-32005 : Unsupported built-in
package sub-procedure is removed */
END;
```

Index

A
ALTIBASE HDB 3

B
Basic Concept 8
Build Step 9, 18
Building the Project 12

C
CLI Mode 3
Command Line Interface (CLI) Mode 14
Compatible Database Systems 3
Configuring the Project 12
Connecting to Source and Destination Databases 12
Create Source and Destination Database Connection 12
Create Source and Destination Database Connections 12
Creating a Project 12

D
Data Type Mapping 29
DB to DB Migration Options 25
DB to File Migration Options 26

E
Executing Run Step in CLI mode 14

G
Getting Started 7
Graphic User Interface (GUI) Mode 10
GUI Mode 3

H
Hardware requirements 3

I
Informix 11.5 to ALTIBASE HDB 34
Installation and Uninstallation 5
Internal Activities 18, 19, 22
Introduction 1

M
Manipulating Data Type Mapping 29
Microsoft SQL Sserver to ALTIBASE HDB 31, 32, 34
Migratable Database Objects 27
Migration Options 25
MySQL to ALTIBASE HDB 32, 34

O
Opening a Project 12
Oracle Database 4
Oracle Database to ALTIBASE HDB 29
Other Databases to ALTIBASE HDB 27
Output 18, 19, 22
Overview 2

P
Prepare Step 8
Project 8
PSM Converter for File 15
Purpose 18, 19, 22

Q
Quick Tour for GUI mode 11

R
Reconcile Step 19
Registering a Database Connection 11
Run Step 9, 22
Running the Project 13

S
Software requirements 3
Starting Migration Center 11
System Requirements 3

T
Terminology 8
The Information pane 10
The Log Pane 11
The Log pane 11
The Project Pane 10
Tools 15

U
Understanding CLI mode 14
Understanding Migration Center 8
Understanding User Interface 10

Z
Zip to Report 15